M9-L1 Problem 1

Here, you will implement three loss functions from scratch in numpy: MAE, MSE, and MAPE.

In [1]:
```
import numpy as np

y_gt1 = np.array([1,2,3,4,5,6,7,8,9,10])
y_pred1 = np.array([1,1.3,3.1,4.6,5.9,5.9,6.4,9.2,8.1,10.5])

y_gt2 = np.array([-3.23594765, -3.74125693, -2.3040903 ,  0.        ,  0.30190142, -1.
y_pred2 = np.array([-3.17886560e+00, -3.72628642e+00, -2.28154027e+00, -2.42424242e-06
```

Mean Absolute Error

Complete the definition for `MAE(y_gt, y_pred)` below. $$ \text{MAE} = \frac{1}{n}\sum_{i=1}^n \left| y_i - \hat{y}_i\right| = \frac{1}{n}\sum_{i=1}^n \left| e_i \right| $$

`MAE(y_gt1, y_pred1)` should return 0.560.

In [2]:
```
def MAE(y_gt, y_pred):
    # YOUR CODE GOES HERE
    n = len(y_gt)
    total = 0
    for i in range(n):
        y = np.abs(y_gt[i] - y_pred[i])
        total+=y
    return (total/n)

print(f"MAE(y_gt1, y_pred1) = {MAE(y_gt1, y_pred1):.3f}")
print(f"MAE(y_gt2, y_pred2) = {MAE(y_gt2, y_pred2):.3f}")
```

```
MAE(y_gt1, y_pred1) = 0.560
MAE(y_gt2, y_pred2) = 0.290
```

Mean Squared Error

Complete the definition for `MSE(y_gt, y_pred)` below. $$ \text{MSE} = \frac{1}{n}\sum_{i=1}^n \left(y_i - \hat{y}_i\right)^2 = \frac{1}{n}\sum_{i=1}^n \left(e_i \right)^2 = \frac{1}{n} e^T e $$

`MSE(y_gt1, y_pred1)` should return 0.454.

In [3]:
```
def MSE(y_gt, y_pred):
    # YOUR CODE GOES HERE
    n = len(y_gt)
    total = 0
    for i in range(n):
        y = np.abs(y_gt[i] - y_pred[i])
        total+=y**2
    return (total/n)
```

```
print(f"MSE(y_gt1, y_pred1) = {MSE(y_gt1, y_pred1):.3f}")
print(f"MSE(y_gt2, y_pred2) = {MSE(y_gt2, y_pred2):.3f}")
```

```
MSE(y_gt1, y_pred1) = 0.454
MSE(y_gt2, y_pred2) = 0.174
```

Mean Absolute Percentage Error

Complete the definition for `MAPE(y_gt, y_pred, epsilon)` below. $$ \text{MAE} = \frac{1}{n}\sum_{i=1}^n \frac{\left| y_i - \hat{y}_i\right|}{ \left| y_i \right| + \varepsilon} = \frac{1}{n}\sum_{i=1}^n \frac{ \left| e_i \right|}{\left| y_i \right| + \varepsilon} $$

`MAPE(y_gt1, y_pred1, 1e-6)` should return 0.112.

In [4]:
```
def MAPE(y_gt, y_pred, epsilon=1e-6):
    # YOUR CODE GOES HERE
    n = len(y_gt)
    total = 0
    for i in range(n):
        num = np.abs(y_gt[i] - y_pred[i])
        denom = np.abs(y_gt[i]) + epsilon
        total+= num/denom
    return (total/n)
print(f"MAPE(y_gt1, y_pred1) = {MAPE(y_gt1, y_pred1):.3f}")
print(f"MAPE(y_gt2, y_pred2) = {MAPE(y_gt2, y_pred2):.3f}")
```

```
MAPE(y_gt1, y_pred1) = 0.112
MAPE(y_gt2, y_pred2) = 0.032
```

In [ ]:
```

```